Third-Party Token-based Authentication and Authorization for Session Initiation Protocol (SIP)
draft-ietf-sipcore-sip-token-authnz-01

Abstract

   This document defines a mechanism for SIP, that is based on the OAuth
   2.0 and OpenID Connect Core 1.0 specifications, to enable the
   delegation of the user authentication and SIP registration
   authorization to a dedicated third-party entity that is separate from
   the SIP network elements that provide the SIP service.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on December 26, 2019.

Table of Contents

1. Introduction

The SIP protocol [RFC3261] uses the framework used by the HTTP
protocol for authenticating users, which is a simple challenge-
response authentication mechanism that allows a server to challenge a
client request and allows a client to provide authentication
information in response to that challenge.

OAuth 2.0 [RFC6749] defines a token based authorization framework to
allow clients to access resources on behalf of their user.

The OpenID Connect 1.0 [OPENID] specifications defines a simple identity layer on top of the OAuth 2.0 protocol, which enables clients to verify the identity of the user based on the authentication performed by a dedicated authorization server, as well as to obtain basic profile information about the user.

This document defines an mechanism for SIP, that is based on the OAuth 2.0 and OpenID Connect Core 1.0 specifications, to enable the delegation of the user authentication and SIP registration authorization to a dedicated third-party entity that is separate from the SIP network elements that provide the SIP service.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.2. SIP User Agent Types

[RFC6749] defines two types of clients, confidential and public, that apply to the SIP User Agents.

o Confidential User Agent: is a SIP UA that is capable of maintaining the confidentiality of the user credentials and any tokens obtained using these user credentials.

o Public User Agent: is a SIP UA that is incapable of maintainings the confidentiality of the user credentials and any obtained tokens.

2. Authentication and Authorization flow

   This flow is used by a Confidential UA with rich UI to authenticate
   to an authorization server and to directly obtain tokens to be able
   to register and get service from the SIP network.

2.1. Overview

   The following sections provide overview of the supported flows.

2.1.1. Configured AS

   The following figure provides a high level view of flow of messages
   when the UA is aware of the AS ahead of time:

```
   UA                            Registrar                          AS
  ---------------------------------------------------------------------
   |                             |                                  |
  [00] The UA prompts the user to provides his credentials          |
   |                             |                                  |
   | [01] HTTP POST /token       |                                  |
   |------------------------------------------------------------------>|
   |                             |                                  |
   |         [02] 200 OK {access_token, refresh_token, [id_token]} |
   |<------------------------------------------------------------------|
   |                             |                                  |
   |                             |                                  |
   | [03] REGISTER               |                                  |
   |      Authorization: Bearer <access_token>                      |
   |----------------------------->|                                  |
   |                             |                                  |
   |                             |       [04] HTTP POST /introspect |
   |                             |              {access_token}      |
   |                             |----------------------------->|
   |                             |                                  |
   |                             |         [05] 200 OK {metadata} |
   |                             |<-----------------------------|
   |                             |                                  |
   |                [06] 200 OK |                                  |
   |<-----------------------------|                                  |
   |                             |                                  |
```
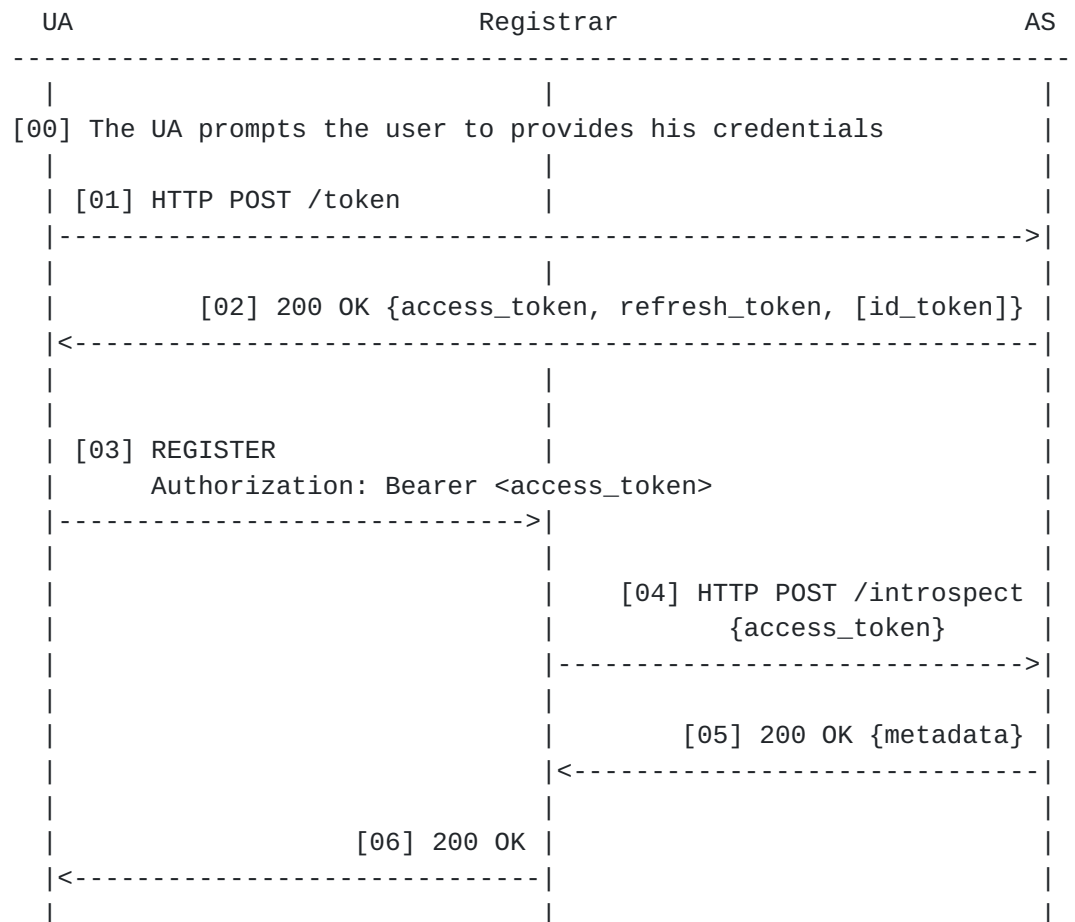
In step [00], the UA collects the user's credentials with the AS.

In steps [01] and [02], the UA first contacts the Authorization
Server to authenticate the user and obtain tokens to be used to get
access to the SIP network.

The tokens returned to the UA depend on the type of server: with an
OAuth Authorization Server, the tokens provided are the access token
and refresh token. With an OpenID Connect server, an additional ID-
Token is returned, which contains the SIP URI of the user. The
method used to authenticate the user and obtain these tokens is out
of scope for this document.

In step [03], the UA starts the registration process with the SIP
registrar by sending a REGISTER request with the access token it
obtained previously.

The registrar validates the access token, and if the access token
provided by the UA is an opaque token, then the registrar MAY perform
an introspection, steps [04] and [05], to obtain more information
about the token and its scope, as per [RFC7662]. Otherwise, after
the registrar validates the token to make sure it was signed by a
trusted entity, it inspects its claims and act upon it.

When the registrar is satisfied with the token, it then replies with
the 200 OK to complete the registration process.

2.1.2. Discovered AS

The following figure provides a high level view of flow of messages
when the UA discovers the AS to conatc from the registrar:

```
   UA                          Registrar                         AS
   ------------------------------------------------------------------------
    |                              |                              |
 [07] The UA prompts the user to provides his credentials        |
    |                              |                              |
    | [08] REGISTER                |                              |
    |----------------------------->|                              |
    |                              |                              |
    | [09] 401 Unauthorized        |                              |
    |      WWW-Authenticate: Bearer "authz_server"="<authz_server>" |
    |<-----------------------------|                              |
    |                              |                              |
    | [10] HTTP POST /token        |                              |
    |-------------------------------------------------------------->|
    |                              |                              |
    |          [11] 200 OK {access_token, refresh_token, [id_token]} |
    |<-------------------------------------------------------------|
    |                              |                              |
    | [12] REGISTER                |                              |
    |      Authorization: Bearer <access_token>                   |
    |----------------------------->|                              |
    |                              | [13] HTTP POST /introspect |
    |                              |         {access_token}       |
    |                              |----------------------------->|
    |                              |                              |
    |                              |      [14] 200 OK {metadata} |
    |                              |<-----------------------------|
    |                              |                              |
    |              [15] 200 OK |                              |
    |<-----------------------------|                              |
    |                              |                              |
```
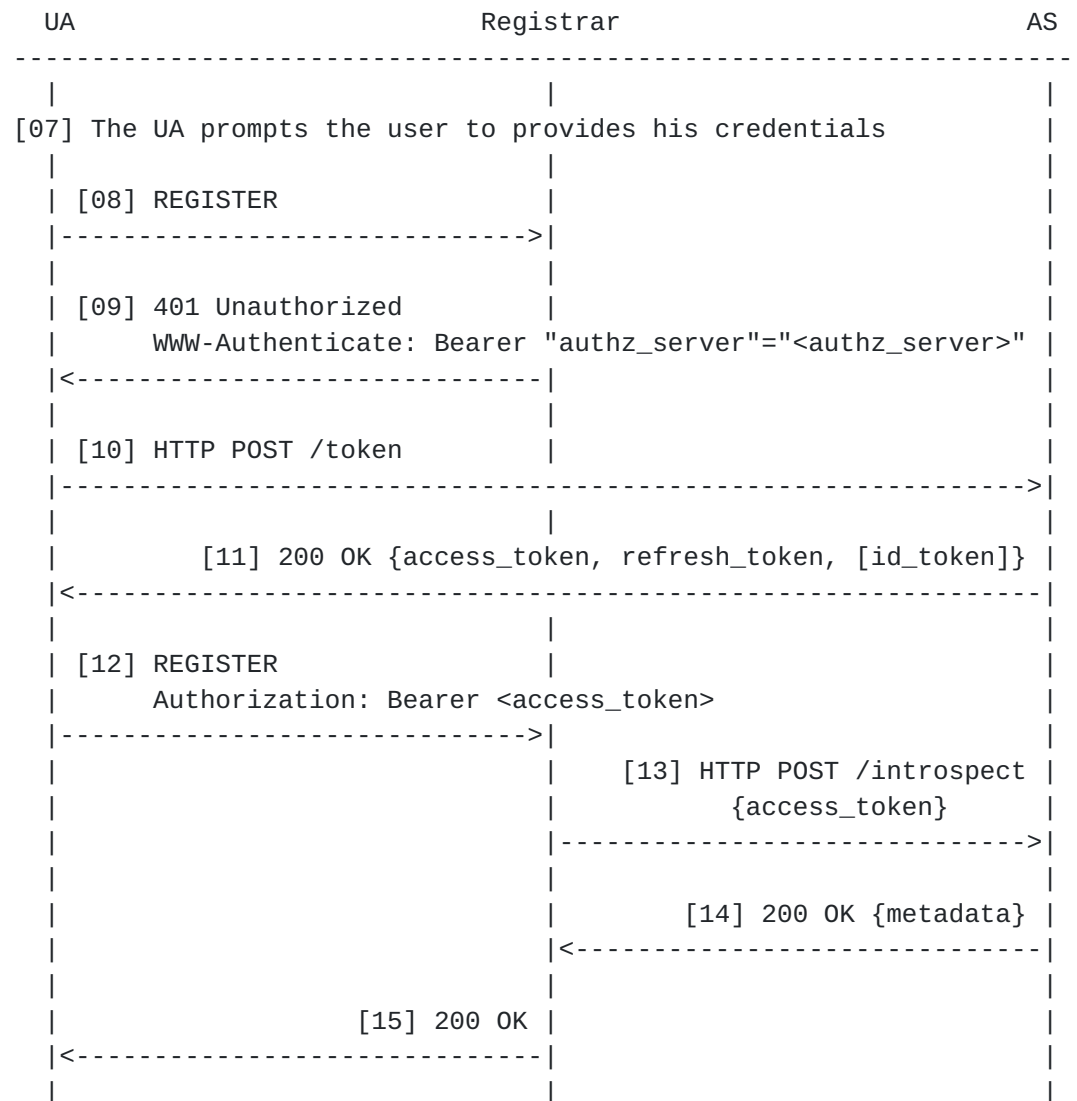
In step [07], the UA collects the user's credentials with the AS.

In step [08] the UA starts the registration process by sending a SIP
REGISTER request to the registrar without any credentials. The
REGISTER request includes an indication that the UA supports token-
based autentication in the form of sip.token media feature tag. The
registrar then challenges the UA, in step [09], by responding with
401 Unauthorized and includes the authorization server to contact to
obtain a token.

In steps [10] and [11], the UA contacts the Authorization Server to authenticate the user and obtain tokens to be used to get access to the SIP network.

The tokens returned to the UA depend on the type of server: with an OAuth Authorization Server, the tokens provided are the access token and refresh token. With an OpenID Connect server, an additional ID-Token is returned, which contains the SIP URI of the user. The method used to authenticate the user and obtain these tokens is out of scope for this document.

In step [12], the UA retries the registration process with the SIP registrar by sending a REGISTER request with the access token it obtained previously.

The registrar validates the access token, and if the access token provided by the UA is an opaque token, then then registrar MAY perform an introspection, steps [13] and [14], to obtain more information about the token and its scope, as per [RFC7662]. Otherwise, after the registrar validates the token to make sure it was signed by a trusted entity, it inspects its claims and act upon it.

2.2. Initial Registration

If the UA has already obtained a token, then the UA starts the registration process, step [03], by sending a REGISTER request, with the access token in the Authorization header, to the registrar.

If the UA does not have a token, then the UA starts the registration process, step [08], by sending a REGISTER request without an Authorization header. The registrar MUST then challenge the UA by responding with 401 Unauthorized and include the WWW-Authenticate Response Header Field which includes the server to contact to obtain a token, as specified in Section 3

The REGISTER request SHOULD include a sip.token media feature tag in the Contact header field of the request, unless it knows (e.g., by means of configuration) that the registrar supports the token authentication mechanism.

The UA MUST include an Authorization header field with the Bearer scheme in the request to carry the access token, as specified in [RFC6750].

When the registrar is satisfied with the token, it then replies with the 200 OK to complete the registration process.

2.3. Subsequent Registrations

All subsequent REGISTER requests from the UA MUST include a valid
access token. The UA MUST obtain a new access token before the
access token expiry period to continue to get service from the
system. The method used to obtain a new fresh access tokens is out
of scope for this document.

The REGISTER request SHOULD include a sip.token media feature tag in
the Contact header field of the request, unless it knows (e.g., by
means of configuration) that the registrar supports the token
authentication mechanism.

3. WWW-Authenticate Response Header Field

This section describes the syntax of the WWW-Authenticate Response
Header Field when used with the Bearer scheme to challenge the UA for
credentials.

```
    challenge  =/  ("Bearer" LWS bearer-cln *(COMMA bearer-cln))
    bearer-cln = realm / domain / scope / authz-server / error /
                 auth-param
    authz-server = "authz_server" EQUAL authz-server-value
    authz-server-value = quoted-string
```

The realm, domain, and auth-param parameters are defined in
[RFC3261].

The scope and error parameters are defined in [RFC6749].

4. 'sip.token' Media Feature Tag

The sip.token media feature tag, when inserted in the Contact header
field of a SIP REGISTER request, conveys that the SIP UA associated
with the tag supports a token based authentication mechanism, where
the user authentication and SIP registration authorization is
performed by a third party. The media feature tag has no values.

token-mt = "+sip.token"

5. Security Considerations

   TODO

6. IANA Considerations

6.1. SIP Media Feaure Tag

6.1.1. sip.token

   This section defines a new media feature tag that extends the "SIP
   Media Feature Tag Registration Tree" subregistry [RFC3840] under the
   "Media Feature Tags" registry (https://www.iana.org/assignments/
   media-feature-tags).

    Media feature tag name: sip.token

    Summary of the media feature indicated by this feature tag: This
        media feature tag, when inserted in the Contact header field
        of a SIP REGISTER request, conveys that the SIP UA associated
        with the tag supports a token based authentication mechanism,
        where the user authentication and SIP registration authorization
        is performed by a third party.

     Values appropriate for use with this feature tag: none

     Related standards or documents: RFC XXXX

     Security considerations: This media feature tag does not introduce
        new security considerations, as it simply indicates support for
        a basic SIP feature. However, if an attacker manages to remove
        the media feature tag from a SIP REGISTER request, the SIP UA
        that inserted it might not be able to authenticate itself with
        the SIP registrar to which the SIP request is addressed, as the
        SIP registrar might not be aware that the SIP UA supports the
        feature associated with the media feature tag.

     Contact: IESG (iesg@ietf.org)

7. Acknowledgments

   The authors would also like to thank Paul Kyzivat for his reviews and
   feedback on this document.

The authors would also like to thank the following for their review
and feedback of the original document that was replaced with this
document:

Andrew Allen, Martin Dolly, Keith Drage, Paul Kyzivat, Jon Peterson,
Michael Procter, Roy Radhika, Matt Ryan, Ivo Sedlacek, Roman Shpount,
Robert Sparks, Asveren Tolga, and Dale Worley.

8. Normative References

[OPENID]   Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and
            C. Mortimore, "OpenID Connect Core 1.0", February 2014.

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119,
            DOI 10.17487/RFC2119, March 1997,
            <https://www.rfc-editor.org/info/rfc2119>.

[RFC3261]   Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
            A., Peterson, J., Sparks, R., Handley, M., and E.
            Schooler, "SIP: Session Initiation Protocol", RFC 3261,
            DOI 10.17487/RFC3261, June 2002,
            <https://www.rfc-editor.org/info/rfc3261>.

[RFC3840]   Rosenberg, J., Schulzrinne, H., and P. Kyzivat,
            "Indicating User Agent Capabilities in the Session
            Initiation Protocol (SIP)", RFC 3840,
            DOI 10.17487/RFC3840, August 2004,
            <https://www.rfc-editor.org/info/rfc3840>.

[RFC6749]   Hardt, D., Ed., "The OAuth 2.0 Authorization Framework",
            RFC 6749, DOI 10.17487/RFC6749, October 2012,
            <https://www.rfc-editor.org/info/rfc6749>.

[RFC6750]   Jones, M. and D. Hardt, "The OAuth 2.0 Authorization
            Framework: Bearer Token Usage", RFC 6750,
            DOI 10.17487/RFC6750, October 2012,
            <https://www.rfc-editor.org/info/rfc6750>.

[RFC7231]   Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer
            Protocol (HTTP/1.1): Semantics and Content", RFC 7231,
            DOI 10.17487/RFC7231, June 2014,
            <https://www.rfc-editor.org/info/rfc7231>.

[RFC7519]   Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token
            (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015,
            <https://www.rfc-editor.org/info/rfc7519>.

   [RFC7523]   Jones, M., Campbell, B., and C. Mortimore, "JSON Web Token
               (JWT) Profile for OAuth 2.0 Client Authentication and
               Authorization Grants", RFC 7523, DOI 10.17487/RFC7523, May
               2015, <https://www.rfc-editor.org/info/rfc7523>.

   [RFC7662]   Richer, J., Ed., "OAuth 2.0 Token Introspection",
               RFC 7662, DOI 10.17487/RFC7662, October 2015,
               <https://www.rfc-editor.org/info/rfc7662>.

Authors' Addresses

   Rifaat Shekh-Yusef (editor)
   Avaya
   425 Legget Drive
   Ottawa, Ontario
   Canada

   Phone: +1-613-595-9106
   EMail: rifaat.ietf@gmail.com

   Christer Holmberg
   Ericsson
   Hirsalantie 11
   Jorvas 02420
   Finland

   EMail: christer.holmberg@ericsson.com

   Victor Pascual
   webrtchacks
   Spain

   EMail: victor.pascual.avila@gmail.com