Workgroup: SIP Core
Internet-Draft:
draft-ietf-sipcore-sip-token-authnz-08
Updates: 3261 (if approved)
Published: 18 February 2020
Intended Status: Standards Track
Expires: 21 August 2020
Authors: R. Shekh-Yusef C. Holmberg V. Pascual
Avaya Ericsson webrtchacks

Third-Party Token-based Authentication and Authorization for Session Initiation Protocol (SIP)

Abstract

This document defines a SIP mechanism that relies on the OAuth 2.0 and OpenID Connect Core 1.0 to enable delegation of the user authentication and SIP registration authorization to a third-party. The document updates RFC 3261.

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 21 August 2020.

Table of Contents

1. Introduction

The Session Initiation Protocol (SIP) [RFC3261] uses the framework used by HTTP [RFC7230] 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 updates [RFC3261], by defining the UAC procedures if it receives a 401/407 response with multiple WWW-Authenticate/Proxy-Authenticate header fields, providing challenges using different authentication schemes for the same realm.

This document defines an mechanism for SIP, that relies 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.

- *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.

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

The mechanism defined in this document MUST only be used with Confidential User Agents, as the UA is expected to obtain and maintain tokens to be able to access the SIP network.

2. SIP Procedures

Section 22 of [RFC3261] defines the SIP procedures for the Digest authentication mechanism procedures. The same procedures apply to the Bearer authentication mechanism, with the changes described in this section.

2.1. UAC Behavior

2.1.1. Obtaining Tokens

When a UAC sends a request without credentials (or with credentials that are no longer valid), and receives a 401 (Unauthorized) or a 407 (Proxy Authentication Required) response that contains a WWW-Authenticate header field (in case of a 401 response) or a Proxy-Authenticate header field (in case of a 407 response) that indicates "Bearer" scheme authentication and contains an address to an Authorization Server, the UAC contacts the Authorization Server in order to obtain tokens, and includes the requested scopes, based on a local configuration.

The tokens returned to the UA depend on the type of AS: with an OAuth AS, the tokens provided are the access token and refresh token. The access token will be sent to the SIP servers to authorize UAC's access to the service. The refresh token will only be used with the AS to get new access token and refresh token, before the expiry of the current access token. With an OpenID Connect server,

an additional ID-Token is returned, which contains the SIP URI and other user specific details, and will be consumed by the UAC.

The detailed OAuth2 procedure to authenticate the user and obtain these tokens is out of scope of this document. [RFC8252] defines procedures for native applications. When using the mechanism defined in [RFC8252] the user will be directed to use a browser for the interaction with the authorization server, allowing the authorization server to prompt the user for multi-factor authentication, redirect the user to third-party identity providers, and the use of single-sign-on sessions.

If the UAC receives a 401/407 response with multiple WWW-Authenticate/Proxy-Authenticate header fields, providing challenges using different authentication schemes for the same realm, the UAC provides credentials for one or more of the schemes that it supports, based on local policy.

NOTE: The address of the Authorization Server might be known to the UAC e.g., using means of configuration, in which case the UAC can contact the Authorization Server in order to obtain the access token before it sends SIP request without credentials.

2.1.2. Protecting the Access Token

[RFC6749] mandates that Access Tokens are protected with TLS when in transit. However, TLS only guarantees hop-to-hop protection when used to protect SIP signaling. Therefore the Access Token MUST be protected in a way so that only authorized SIP servers will have access to it. Endpoints that support this specification MUST support encrypted JSON Web Tokens (JWT) [RFC7519] for encoding and protecting Access Token when included in SIP requests, unless some other mechanism is used to guarantee that only authorized SIP endpoints have access to the Access Token.

2.1.3. REGISTER Request

The procedures in this section assumes that the UAC has obtained a token as specified in section Section 2.1.1

When the UAC sends a REGISTER request after it received a challenge containing the Bearer scheme, then to resolve that particular challenge it needs to send a request with an Authorization header field containing the response to that challenge, including the Bearer scheme carrying a valid access token in the request, as specified in [RFC6750].

Note that if there were multiple challenges with different schemes then it maybe able to successfully retry the request using non-Bearer credentials.

Based on local policy, the UAC MAY include an access token that has been used for another binding associated with the same AOR in the request.

If the access token included in a REGISTER request is not accepted, and the UAC receives a 401 response or a 407 response, the UAC follows the procedures in Section 2.1.1.

2.1.4. Non-REGISTER Request

The procedures in this section assumes that the UAC has obtained a token as specified in section Section 2.1.1

When a UAC sends a request, after it received a challenge containing the Bearer scheme, then the UAC MUST include an Authorization header field with a Bearer scheme, carrying a valid access token in the request, as specified in [RFC6750]. Based on local policy, the UAC MAY include an access token that has been used for another dialog, or for another stand-alone request, if the target of the new request is the same.

If the access token included in a request is not accepted, and the UAC receives a 401 response or a 407 response, the UAC follows the procedures in Section 2.1.1.

2.2. UAS and Registrar Behavior

When a UAS or Registrar receives a request that fails to contain authorization credentials acceptable to it, it SHOULD challenge the request by sending a 401 (Unauthorized) response. To indicate that it is willing to accept an OAuth2 token as a credential the UAS/Registrar MUST include a Proxy-Authentication header field in the response, indicate "Bearer" scheme and include an address of an Authorization Server from which the originator can obtain an access token.

When a UAS/Registrar receives a SIP request that contains an Authorization header field with an access token, the UAS/Registrar MUST validate the access token, using the procedures associated with the type of access token used, e.g. [RFC7519]. If the validation is successful the UAS/Registrar can continue to process the request

using normal SIP procedures. If the validation fails, the UAS/ Registrar MUST reject the request.

2.3. Proxy Behavior

When a proxy receives a request that fails to contain authorization credentials acceptable to it, it SHOULD challenge the request by sending a 407 (Proxy Authentication Required) response. To indicate that it is willing to accept an OAuth2 token as a credential the proxy MUST include a Proxy-Authentication header field in the response, indicating "Bearer" scheme and including an address to an Authorization Server from which the originator can obtain an access token.

When a proxy wishes to authenticate a received request, it MUST search the request for Proxy-Authorization header fields with 'realm' parameters that match its realm. It then MUST successfully validate the credentials from at least one Proxy-Authorization header field for its realm. When the scheme is Bearer the proxy MUST validate the access token, using the procedures associated with the type of access token used, e.g. [RFC7519].

3. Access Token Claims

The type of services that an access token grants access to can be determined using different methods. Which methods are used and the granted access provided by the token is based on local policy agreed between the AS and the registrar.

If an access token is encoded as a JWT, it might contain a list of claims [RFC7519], some registered and some are application specific claims. The REGISTRAR can grant access to services either based on such claims, using some other mechanism, or a combination of claims and some other mechanism. If an access token is a reference token, the REGISTRAR will grant access based on some other mechanism. Examples of such other mechanisms are introspection [RFC7662], user profile lookups, etc.

4. 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, by extending the 'challnge' header field defined by [RFC3261].

```
challenge  =/  ("Bearer" LWS bearer-cln *(COMMA bearer-cln))
bearer-cln = realm / scope / authz-server / error /
             auth-param
authz-server = "authz_server" EQUAL authz-server-value
authz-server-value = https-URI
realm = <defined in RFC3261>
auth-param = <defined in RFC3261>
scope = <defined in RFC6749>
error = <defined in RFC6749>
https-URI = <defined in RFC7230>
```

Figure 1: Bearer Scheme Syntax

The authz-server parameters contains the HTTPS URI, as defined in [RFC7230], of the authorization server. The UA can discover metadata about the AS using a mechanism like the one defined in [RFC8414].

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

As per [RFC3261], the realm string alone defines the protection domain. [RFC3261] states that the realm string must be globally unique and recommends that the realm string contains a hostname or domain name. It also states that the realm string should be human-readable identifier that can be rendered to the user.

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

The scope parameter could be used by the registrar/proxy to indicate to the UAC the minimum scope that must be associated with the access token to be able to get service. As defined in [RFC6749], the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings. The strings are defined by the authorization server. The values of the scope parameter is out of scope of this document. The UAC will use the scope provided by the registrar to contact the AS and obtain a proper token with the requested scope.

The error parameter could be used by the registrar/proxy to indicate to the UAC the reason for the error, with possible values of "invalid_token" or "invalid_scope".

5. Example Flows

5.1. Registration

The figure below shows an example of a SIP registration, where the UA is informed about the Authorization Server (AS) from where to

obtain an access token by the registratar in a 401 response to the REGISTER request.

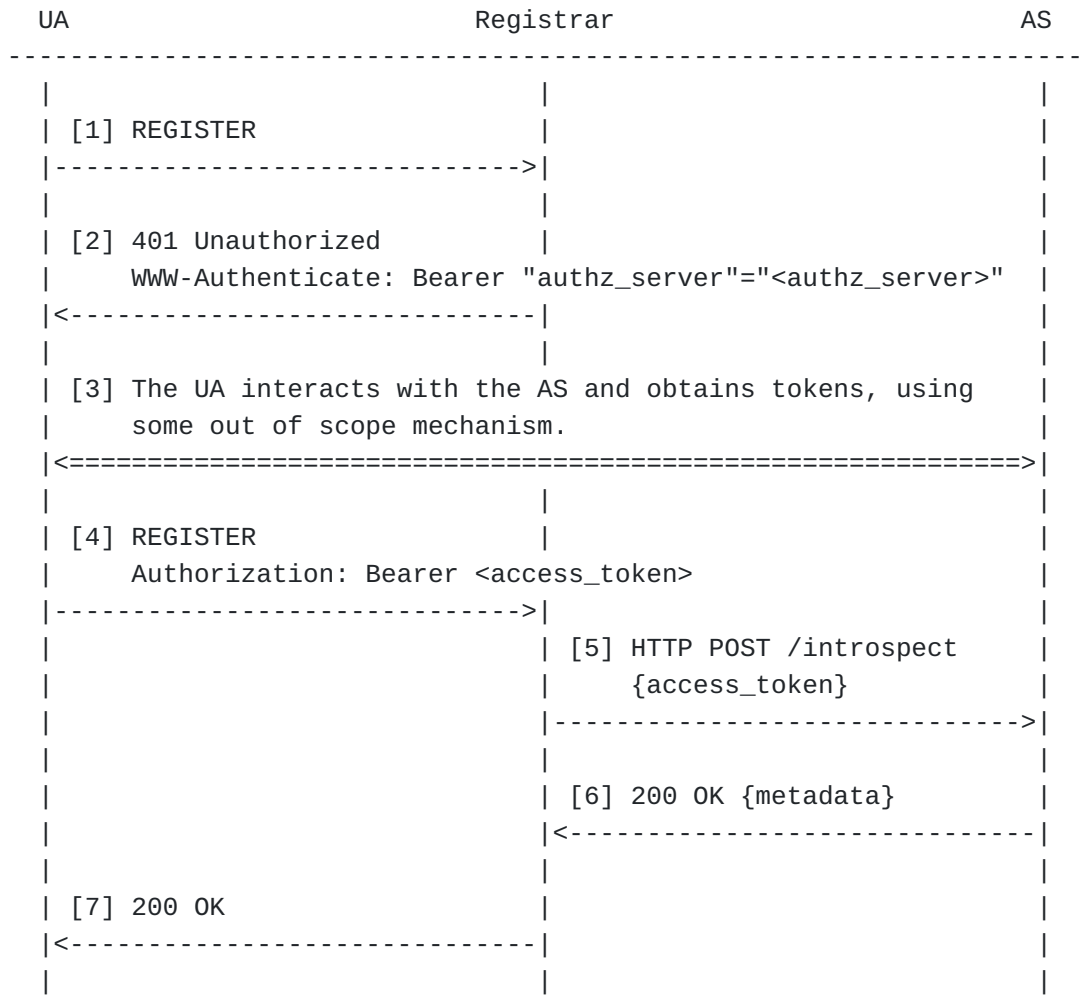

Figure 2: Example Registration Flow

In step [1], the UA starts the registration process by sending a SIP REGISTER request to the registrar without any credentials.

In step [2], the registrar challenges the UA, by sending a SIP 401 (Unauthorized) response to the REGISTER request. In the response the registrar includes information about the AS to contact in order to obtain a token.

In step [3], the UA interacts with the AS, potentially using the OAuth Native App mechanism defined in [RFC8252], authenticates the user and obtains the tokens needed to access the SIP service.

In step [4], the UA retries the registration process by sending a new SIP REGISTER request that includes the access token that the UA obtrained previously.

The registrar validates the access token. If the access token is a reference token, the registrar MAY perform an introspection, as in steps [5] and [6], in order to obtain more information about the access 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.

In step [7], once the registrar has succesfully verified and accepted the access token, it sends a 200 (OK) response to the REGISTER request.

5.2. Registration with Pre-Configured AS

The figure below shows an example of a SIP registration, where the UA has pre-configured information about the Authorization Server (AS) from where to obtain the access token.

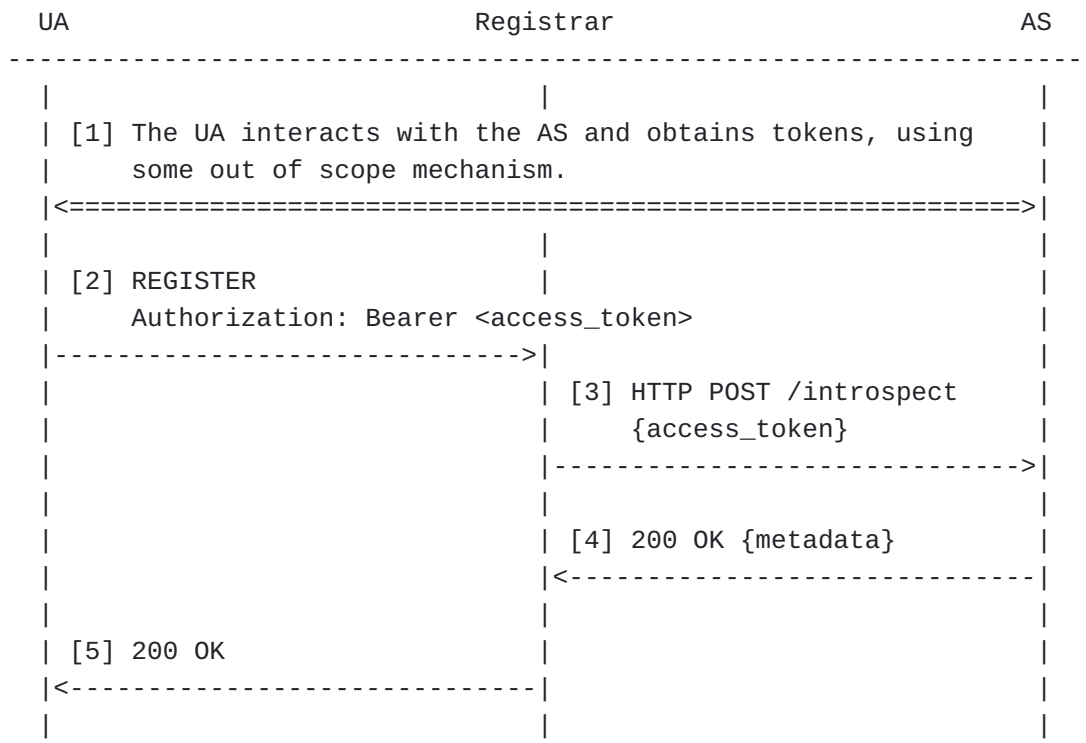

Figure 3: Example Registration Flow - Authorization Server Information Preconfigured

In step [1], the UA interacts with the AS, potentially using the OAuth Native App mechanism defined in [RFC8252], authenticates the user and obtains the tokens needed to access the SIP service.

In step [2], the UA retries the registration process by sending a new SIP REGISTER request that includes the access token that the UA obtrained previously.

The registrar validates the access token. If the access token is a reference token, the registrar MAY perform an introspection, as in steps [3] and [4], in order to obtain more information about the access 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.

In step [5], once the registrar has succesfully verified and accepted the access token, it sends a 200 (OK) response to the REGISTER request.

6. Security Considerations

The security considerations for OAuth are defined in [RFC6749]. The security considerations for bearer tokens are defined in [RFC6750]. The security considerations for JSON Web Tokens (JWT) are defined in [RFC7519]. These security considerations also apply to SIP usage of access token as defined in this document.

[RFC6749] mandates that Access Tokens are protected with TLS. However, TLS only guarantees hop-to-hop protection when used to protect SIP signaling. Therefore the Access Token MUST be protected in a way so that only authorized SIP endpoints will have access to it. Endpoints that support this specifications MUST support encrypted JSON Web Tokens (JWT) [RFC7519] for encoding and protecting Access Token when included in SIP requests, unless some other mechanism is used to guarantee that only authorized SIP endpoints have access to the Access Token.

7. IANA Considerations

8. Acknowledgments

The authors would like to specially thank Paul Kyzivat for his multiple detailed reviews and suggested text that significanly improved the quality of the document.

The authors would also like to thank the following for their review and feedback on this document:

Olle Johansson, Roman Shpount, Dale Worley, and Jorgen Axell.

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.

The authors would also like to thank Jean Mahoney for her review, editorial help, and the coversion of the XML source file from v2 to v3.

9. 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>.

[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>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and

Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[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>.

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

[RFC8252] Denniss, W. and J. Bradley, "OAuth 2.0 for Native Apps", BCP 212, RFC 8252, DOI 10.17487/RFC8252, October 2017, <https://www.rfc-editor.org/info/rfc8252>.

[RFC8414] Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/info/rfc8414>.

Authors' Addresses

Rifaat Shekh-Yusef
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