

SIP Core
Internet-Draft
Updates: [3261](#) (if approved)
Intended status: Standards Track
Expires: August 9, 2018

R. Shekh-Yusef, Ed.
Avaya
C. Holmberg
Ericsson
V. Pascual
webprtchacks
February 5, 2018

Third-Party Authentication for Session Initiation Protocol (SIP)
draft-ietf-sipcore-sip-authn-02

Abstract

This document defines an authentication 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 to a dedicated third-party IdP 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 August 9, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must

Internet-Draft

Third-Party Authentication for SIP

February 2018

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	3
1.1.	Terminology	3
1.2.	Roles	3
1.3.	ID Token	4
1.4.	SIP User Agent Types	5
1.5.	Authentication Types	5
2.	Authentication using the Authorization Code Flow	6
2.1.	Public UA with Rich UI	6
2.1.1.	Initial Registration	7
2.1.2.	Shared-Key	8
2.1.3.	Subsequent Registration	8
2.1.4.	Token Refresh	9
2.2.	Public UA with Limited UI	10
2.2.1.	Registration	10
2.2.2.	Shared-Key	11
2.2.3.	Token Refresh	11
2.2.4.	Subsequent Registration	12
3.	Authentication using the Resource Owner Password Credentials flow	13
3.1.	Overview	13
3.2.	Initial Registration	13
3.3.	Subsequent Requests	14
4.	Authorization Header Syntax	14
5.	Security Considerations	14
6.	IANA Considerations	15

6.1.	Shared Key Feature-Capability Indicator	15
7.	Acknowledgments	15
8.	References	16
8.1.	Normative References	16
8.2.	Informative References	16

Authors' Addresses	16
------------------------------	--------------------

[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 IdP entity, as well as to obtain basic profile information about the user.

This document defines an authentication 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 to a dedicated third-party IdP 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.](#) Roles

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.

In a typical SIP network, it is the management element in the system that acts as a resource owner.

resource server

The server hosting the protected resources or services, capable of accepting and responding to protected resource and services requests using access tokens.

OAuth 2.0 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).

SIP client

An application making requests to access SIP services on behalf of the end-user.

authorization server

The server issuing tokens to the OAuth 2.0 client or SIP Client after successfully authenticating the resource owner and obtaining authorization.

Identity Provider (IdP)

This definition is borrowed from [[MITKB](#)]

"IdP (Identity Provider), is a system that creates, maintains, and manages identity information for principals (users, services, or systems) and provides principal authentication to other service providers (applications) within a federation or distributed network. It is a trusted third party that can be relied upon by users and servers when users and servers are

establishing a dialog that must be authenticated. The IdP sends an attribute assertion containing trusted information about the user to the SP".

[1.3.](#) ID Token

ID token, as defined in the OpenID document, is a security token that contains claims about the authentication of an end-user by an authorization server.

[1.4.](#) 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 maintaining the confidentiality of the user credentials and any obtained tokens.

[1.5.](#) Authentication Types

There are two types of user authentications in SIP:

- o Proxy-to-User: which allows a server that is providing a service to authenticate the identity of a user before providing the service.

- o **User-to-User:** which allows a user receiving a request to authenticate the identity of the remote user before processing the request.

The mechanism defined in this document addresses the proxy-to-user authentication only. For user-to-user authentication refer to the mechanism defined in [[RFC474bis](#)].

[2.](#) Authentication using the Authorization Code Flow

Authorization Code Flow is used by the SIP UA to authenticate to a third-party IdP entity and to obtain an authorization code that would be later used by the SIP Proxy to obtain tokens to allow the SIP UA to register and get service from the SIP network.

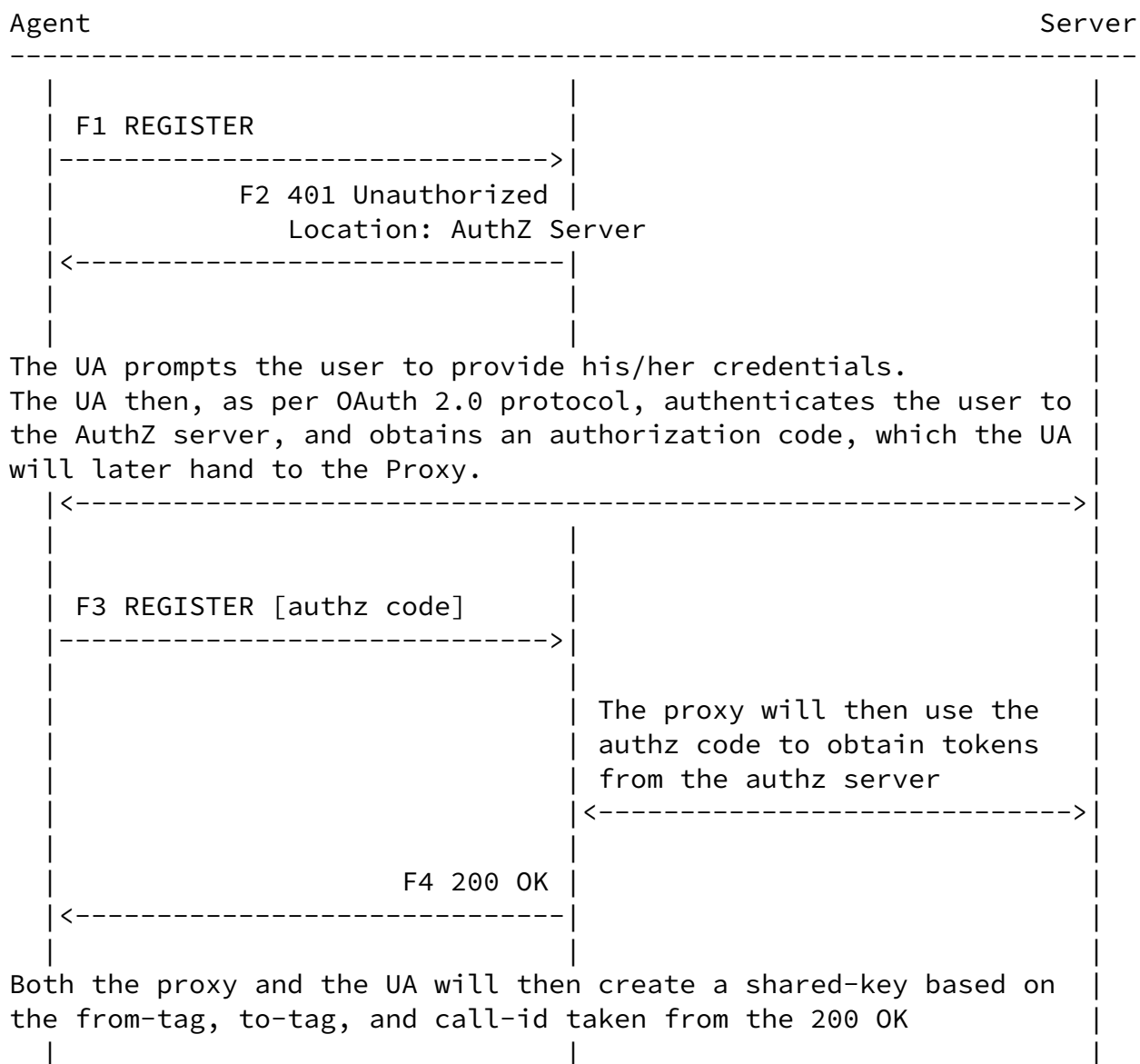
[2.1.](#) Public UA with Rich UI

The following figure provides a high level view of flow of messages for the user authentication using a Public UA that has a rich UI that would prompt the user for his credentials:

User

Proxy

Authorization



The UA initially sends a REGISTER request (F1) without providing any credentials. The proxy redirects the UA by responding with 401 Unauthorized (F2).

The UA will then contact the Authorization Server and obtain an authorization code to be used with the SIP proxy.

The UA then retries the request (F3) and includes the authorization

code in the body of the request.

The proxy then contacts the Authorization Server and exchanges the authorization code for tokens. If the proxy is successful in exchanging the authorization code with the tokens, the proxy then replies with 200 OK to complete the registration process, and locally generates the shared-key with the UA for this user.

When the UA receives the 200 OK, it will follow the same procedure used by the proxy and calculate its shared-key locally.

2.1.1. Initial Registration

The UA initiates the process by sending a REGISTER request (F1) to the proxy. The proxy will redirect the UA to the Authorization Server by responding with 401 Unauthorized (F2) that includes the address of the Authorization Server in the form of an HTTP URI in a Location header field, as defined in [\[RFC7231\], section 7.1.2.](#)

The UA will then contact the Authorization Server and obtain an authorization code to be used with the SIP proxy. The method used by the UA to obtain the code is out of scope for this document.

The UA will then send a new REGISTER request (F3) and include the authorization code, using the "application/x-www-form-urlencoded" format, in the body of the request with the following parameters:

grant_type (REQUIRED)

Value MUST be set to "authorization_code".

code (REQUIRED)

The authorization code received from the authorization server.

The proxy then contacts the Authorization Server and exchanges the authorization code for access token, refresh token, and maybe ID

token. The method used by the UA to obtain the tokens is out of

scope for this document.

If the proxy is successful in exchanging the authorization code with the tokens, the proxy then responds with 200 OK (F4) to the UA to complete the registration process; otherwise, the proxy MUST reply with 401 Unauthorized.

[2.1.2.](#) Shared-Key

The shared-key could be used to allow the UA to recover from a connection loss to the server without the need to prompt the user for credentials.

If the server supports the use of shared-key, it MUST indicate that by adding the new sip.shared-key parameter to the feature-caps header in the 200 OK response to the REGISTER request.

After sending the 200 OK to the UA to complete the registration process, assuming that both the server and the client support this feature, the proxy and the UA use the HMAC-SHA256(key, message) to calculate the shared-key associated with this user as follows:

key

The authorization code obtained from the Authorization Server.

message

The concatenation of the 'from-tag', 'to-tag', and 'call-id' of the 200 OK that completes the registration process.

[2.1.3.](#) Subsequent Registration

When the UA loses its connection to the proxy and it wants to send a new registration request to the proxy, the UA will send a new REGISTER request and include the proof-of-possession (pop) of the shared-key in the body of the request, using the "application/x-www-form-urlencoded" format:

grant_type (REQUIRED)

Value MUST be set to "proof_of_possession".

pop (REQUIRED)

The pop calculated using the shared-key created the first time the UA registered with the proxy.

The pop is calculated using the shared-key as follows:

$$\text{pop} = \text{HMAC-SHA256}(\text{shared-key}, \text{digest-string})$$

See [\[RFC4474\]](#), [section 9](#), for the SIP headers to hash to create digest-string.

If the server supports the pop mechanism, then the server must validate the pop provided by the client. If the validation is successful, the server MUST reply with a 200 OK to complete the registration process; otherwise, the server MUST reply with 401 Unauthorized.

[2.1.4](#). Token Refresh

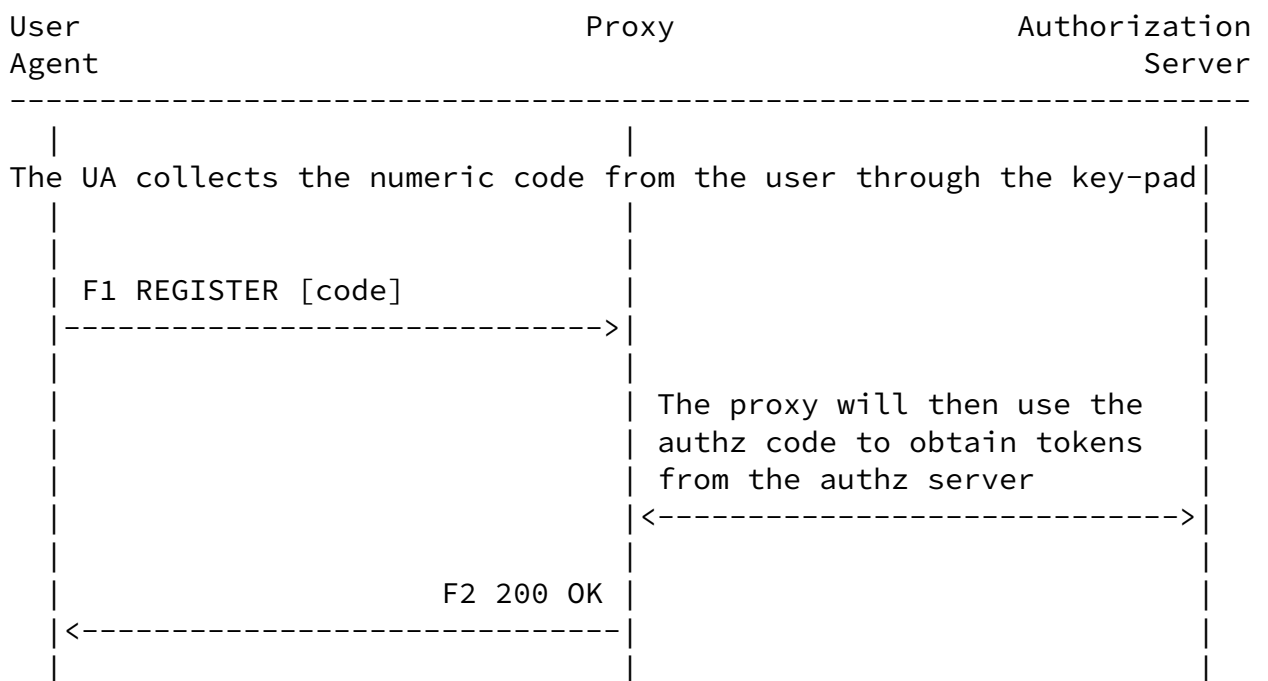
Before the tokens expire, the proxy makes a refresh request to the Authorization Server to try to obtain new tokens. The method used by the UA to refresh the tokens is out of scope for this document.

If the proxy fails to refresh the tokens, then it MUST challenge the next request from the UA, and as a result the UA MUST go through the authorization process again.

[2.2.](#) Public UA with Limited UI

The following figure provides a high level view of flow of messages for the user authentication using a Public UA that has a limited UI that cannot prompt the user for his credentials.

This use case requires the user to use an out-of-band mechanism (e.g. a Browser or a One-Time-Password (OTP) application) to authenticate to the Authorization Server and obtain a short lived numeric authorization code that would be used by the phone to register with the SIP proxy.



[2.2.1.](#) Registration

The UA will send a REGISTER request (F1) and include the code in the body of the request, using the "application/x-www-form-urlencoded" format, with the following parameters:

grant_type (REQUIRED)

Value MUST be set to "authorization_code".

code (REQUIRED)

The code received from the authorization server through the out-of-bound mechanism.

The proxy then contacts the Authorization Server and exchanges the authorization code for access token, refresh token, and maybe an ID token. The method used by the UA to obtain the tokens is out of scope for this document.

If the proxy is successful in exchanging the authorization code with the tokens, the proxy then responds with 200 OK (F2) to the UA to complete the registration process; otherwise, the proxy MUST reply with 401 Unauthorized.

[2.2.2.](#) Shared-Key

The shared-key could be used to allow the UA to recover from a connection loss to the server without the need to prompt the user for credentials.

If the server supports the use of shared-key, it MUST indicate that by adding the new sip.shared-key parameter to the feature-caps header in the 200 OK response to the REGISTER request.

After sending the 200 OK to the UA to complete the registration process, assuming that both the server and the client support his feature, the proxy and the UA use the HMAC-SHA256(key, message) to calculates the shared-key associated with this user as follows:

key

The authorization code obtained from the Authorization Server.

message

The concatenation of the 'from-tag', 'to-tag', and 'call-id' of the 200 OK that completes the registration process.

[2.2.3.](#) Token Refresh

Before the tokens expire, the proxy makes a refresh request to the Authorization Server to try to obtain new tokens. The method used by the UA to refresh the tokens is out of scope for this document.

If the proxy fails to refresh the tokens, then it MUST challenge the next request from the UA, and as a result the UA MUST go through the authorization process again.

[2.2.4.](#) Subsequent Registration

When the UA loses its connection to the proxy and it wants to send a new registration request to the proxy, the UA will send a new REGISTER request and include a proof-of-possession (pop) of the shared-key in the body of the request, using the "application/x-www-form-urlencoded" format:

grant_type (REQUIRED)

Value MUST be set to "proof_of_possession".

pop (REQUIRED)

The pop calculated using the shared-key created the first time the UA registered with the proxy.

The pop is calculated using the shared-key as follows:

pop = HMAC-SHA256(shared-key, digest-string)

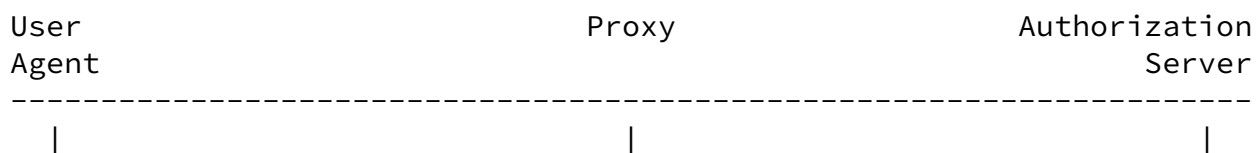
See [\[RFC4474\]](#), [section 9](#), for the SIP headers to hash to create digest-string.

[3.](#) Authentication using the Resource Owner Password Credentials flow

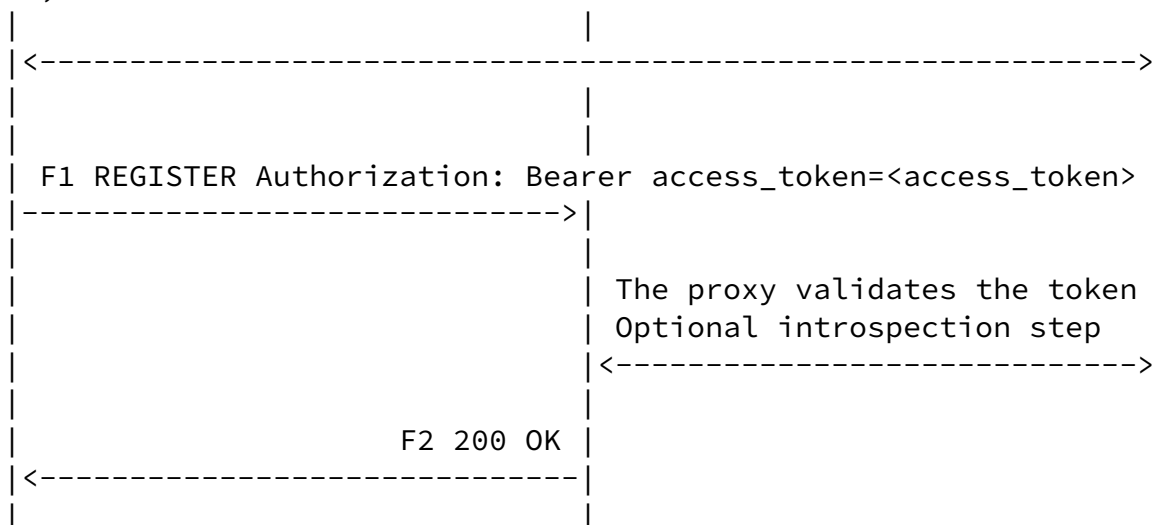
The resource owner password credentials flow is used by a Confidential UA with rich UI to authenticate to a third-party IdP entity and to directly obtain tokens to be able to register and get service from the SIP network.

[3.1.](#) Overview

The following figure provides a high level view of flow of messages for the OAuth Resource Owner Password Credentials flow:



The UA contacts the authorization server and authenticates the user, and as a result obtains an access and refresh tokens.



[3.2.](#) Initial Registration

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 method used by the UA to obtain the tokens is out of scope for this document.

The UA starts the registration process with the SIP proxy by sending a REGISTER request (F1) with the access token it obtained previously.

The UA includes an Authorization header field with the Bearer scheme in the request to carry the access token obtained previously.

The proxy then validates the token, and MAY perform an introspection step to get more information about the token and its scope. The introspection step is out of scope for this document.

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

[3.3.](#) Subsequent Requests

All subsequent 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.

[4.](#) Authorization Header Syntax

This section describes the syntax of the authorization header with the Bearer scheme.

```
Authorization = "Authorization" HCOLON "Bearer" LWS
                "access_token" EQUAL access_token
access_token = quoted-string
```

[5.](#) Security Considerations

As this document uses the mechanism defined in the OAuth 2.0 [[RFC6749](#)], many of the security consideration in the OAuth 2.0 document apply to this document too.

The shared-key mechanism used with the Public UA allows a UA to re-register without the need to obtain a new access code. If this shared-key is leaked, an adversary will be able to register a UA and impersonate the attacked user.

To reduce the chances of the shared-key being leaked, the UA should not store the shared-key in a permanent storage, but keep it in memory only.

A server should limit the use of shared-key to clients that are able to provide an adequate level of protection for the shared-key. In some deployments, the server might decide not to support the use of shared-key at all.

[6.](#) IANA Considerations

[6.1.](#) Shared Key Feature-Capability Indicator

This document defines the feature capability sip.shared-key in the "SIP Feature-Capability Indicator Registration Tree" registry defined in [[RFC6809](#)].

Name: sip.shared-key

Description: This feature-capability indicator, when included in a Feature-Caps header field of a REGISTER response, indicates that the server supports the use of shared-key mechanism.

Reference: [this document]

7. Acknowledgments

The authors would like to thank the following for their review and feedback:

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.

Special thanks to Jon Peterson for a long discussion on the ideas and concepts around the use of OpenID/OAuth as an authentication and authorization mechanisms in a SIP network.

8. References

8.1. Normative References

- [MITKB] "IdP (Identity Provider)", MIT Knowledge Base, <http://kb.mit.edu/confluence/x/XoK2>, March 2011.
- [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", [RFC 2119](#), March 1997.
- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, H., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", [RFC 3261](#), June 2002.
- [RFC4474] Peterson, J. and C. Jennings, "Enhancements for Authenticated Identity Management in the Session Initiation Protocol (SIP)", August 2006.
- [RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", [RFC 6749](#), October 2012.
- [RFC7231] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", [RFC 7231](#), June 2014.
- [RFC7662] Richer, J., "OAuth 2.0 Token Introspection", [RFC 7662](#), October 2015.

8.2. Informative References

- [RFC474bis]
Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in SIP", <https://datatracker.ietf.org/doc/draft-ietf-stir-rfc4474bis/>, February 2017.

Authors' Addresses

Internet-Draft

Third-Party Authentication for SIP

February 2018

Rifaat Shekh-Yusef (editor)

Avaya
250 Sidney Street
Belleville, Ontario
Canada

Phone: +1-613-967-5176
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

